

# Human Leader and Robot Follower Team: Correcting Leader's Position from Follower's Heading

Johann Borenstein<sup>1\*</sup>, David Thomas<sup>1</sup>, Brandon Sights<sup>2</sup>,  
Lauro Ojeda<sup>1</sup>, Peter Bankole<sup>2</sup>, Donnie Fellars<sup>2</sup>

<sup>1</sup> University of Michigan, Dept of Mechanical Engineering, 2260 Hayward St, Ann Arbor, MI 48109. \*POC: johannb@umich.edu, ph.: 734-763-1560

<sup>2</sup> Space and Naval Warfare Systems Center, San Diego, 53560 Hull Street, San Diego, CA 92152

## ABSTRACT

In multi-agent scenarios, there can be a disparity in the quality of position estimation amongst the various agents. Here, we consider the case of two agents – a leader and a follower – following the same path, in which the follower has a significantly better estimate of position and heading. This may be applicable to many situations, such as a robotic “mule” following a soldier. Another example is that of a convoy, in which only one vehicle (not necessarily the leading one) is instrumented with precision navigation instruments while all other vehicles use lower-precision instruments. We present an algorithm, called Follower-derived Heading Correction (FDHC), which substantially improves estimates of the leader's heading and, subsequently, position. Specifically, FDHC produces a very accurate estimate of heading errors caused by slow-changing errors (e.g., those caused by drift in gyros) of the leader's navigation system and corrects those errors.

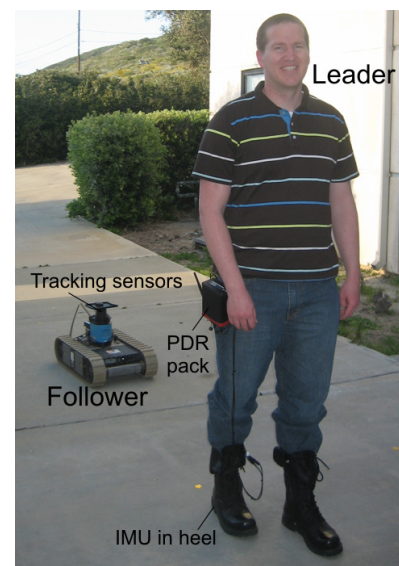
**Keywords:** Unmanned Ground Vehicle (UGV), robot, leader-follower, dead-reckoning, gyro, drift, error correction

## 1 INTRODUCTION

Leader-follower constellations have been subject of research for many years now. Most of the work in this area focuses on control aspects related to steering an unmanned follower vehicle so that it follows a lead vehicle [1], [2], [3]. Other work focuses on sensors and algorithms to enable follower behaviors. Typically, followers use one of two types of sensors: (1) Breadcrumb trail tracking [4] or (2) Direct line-of-sight sensors [5]. Typical sensors for the former are scanning laser rangefinders and computer vision. Typical sensors for the latter are GPS, Inertial Measurement Units (IMUs) or single axis gyros, magnetometers and odometry.

Hogg et al. [4] describe a system, in which a pair of small mobile robots (both were PackBots from iRobot [6]) made up the leader-follower team. Both the leader and the follower were equipped with Differential GPS (DGPS), inertial navigation sensors (INS), and a compass for navigation, as well as LADAR and stereo vision for obstacle avoidance. Their system was designed not to require line-of-sight at all. During motion, the leader would produce a breadcrumb trail of absolute waypoints, which was then tracked by the follower. Because of the absolute position estimates provided by GPS, position errors were bounded and their leader-follower team could conceivably travel indefinitely. While their system can tolerate short interruptions of GPS coverage, during longer outages position errors accrued by both the leader and the follower would likely cause the system to fail.

Among the direct line-of-sight-based sensor system approaches, the system developed at SPAWAR San Diego (see Figure 1) stands out because of its robust fusing of three sensor modalities: LADAR, a thermal camera, and a monocular color



**Figure 1:** Leader and follower during an experiment at SPAWAR.

camera [5]. The thermal camera is particularly useful when the leader is human as is the case in our experiments, although our proposed method is also suitable for robotic leaders.

One problem for line-of-sight-based systems is the occasional interruption of the line-of-sight. If the interruption persists for more than a few seconds, it becomes necessary to switch to breadcrumb trail-style leader-follower behavior. For that eventuality, it is necessary to equip both the leader and the follower with dead-reckoning instrumentation. A greater problem exists, though, when the leader is human. In order to facilitate non-line-of-sight following of the human leader, the leader needs to be equipped with an accurate position estimation system that is similar in accuracy to what can be expected from a robotic leader equipped with dead-reckoning instrumentation. However, very few tracking devices exist (e.g., Honeywell DRM-4000) that can track the position of a walking person in GPS denied environments. All of the existing IMU-based pedestrian tracking systems suffer from large drift rates in the (usually MEMS-based) gyros, which cause unbounded and rapid growth of heading errors. They can fill in line-of-sight disruptions of a few seconds, but not for extensive line-of-sight outages lasting more than, say, five minutes. We believe that there is significant value in a leader-follower system that does not only tolerate occasional line-of-sight outages, but rather allows predominantly breadcrumb-trail style navigation and requires only occasional and brief line-of-sight updates.

In order to achieve this goal, we developed a method, by which the robot follower (or “follower,” in short) actively helps correct the leader’s heading errors. To this end, the follower is equipped with a high-accuracy position estimation system that uses a fiber optic gyro and Simultaneous Localization and Mapping (SLAM) capabilities. During leader-follower operation, the follower continuously streams its own accurate heading and position estimates to the leader. Our proposed Follower-derived Heading Correction (FDHC) algorithm, running on the leader’s computer, uses the follower’s streaming accurate heading data to estimate and correct for its own drift-induced heading errors and, consequently, reduce position errors. A surprising quality of our method is that the follower can lag 30 seconds or more behind the leader, and still correct the leader’s heading errors. The result of implementing our method is that the leader’s position estimates become far more accurate, thereby allowing extended periods of leader-follower operation without line-of-sight.

The conditions for our approach to work are:

1. The follower generally follows the leader’s route but may perform local obstacle avoidance or other localized maneuvers that differ from local maneuvers of the leader.
2. The follower has means for measuring bearing and distance to the leader when line-of-sight conditions exist, but these measurements are needed only occasionally, say, every five minutes and even then only for a few seconds.
3. The nominal distance between the leader and the follower is short, on the order of 1-50 meters (about 1-30 seconds in terms of time). In the experiments described in this paper, however, the distance was very short, about 0.5-2 m, due to range limitations of the line-of-sight sensors employed on the follower.
4. The leader and the follower have suitable bi-directional low-bandwidth RF data communication capability.

While the method presented in this paper was developed specifically for the above described human leader–robot follower constellation, it can easily be applied to other leader-follower constellations, including the special case of convoys. For example, in convoys it might be cost effective to use low-quality navigation instruments on most of the vehicles, but have at least one vehicle equipped with high-quality navigation instruments. Using our proposed approach, the vehicle with the high-quality instruments can correct heading errors of all other vehicles, regardless of their position in the convoy and with only occasional need for line-of-sight.

## **2 THE FOLLOWER-DERIVED HEADING CORRECTION (FDHC) METHOD**

For convenience, we introduce two simplifying abstractions: (1) the virtual Z-axis gyro and (2) slow-changing heading errors. We define these abstractions before discussing our FDHC algorithm in detail.

### Virtual Z-axis gyro

Our Personal Dead-reckoning (PDR) system (described in some more detail in Section 3.1.) estimates position and attitude of the walker in six degrees-of-freedom (DOF). While the data from the foot-mounted Inertial Measurement Unit (IMU) of the PDR system requires complex mathematical procedures and transformations, in the end the system’s output is reduced to a series of x-y positions at footfall intervals. From successive positions one can easily compute the change

of direction of travel from footfall to footfall, and from that the rate of rotation around the world Z-axis. Therefore, we introduce the notion of what we call a “virtual Z-axis gyro,” The output of this virtual Z-axis gyro, or just “gyro,” in short, can be the output of any position tracking system, expressed in terms of rate of rotation around the world Z-axis. The errors in the virtual Z-axis gyro may be difficult to model as a function of the errors of the individual physical gyros of an IMU. However, as we will see, our method does not require the definition of a model. Heading is estimated based on the integration of rates of turn “measured” by the virtual z-axis gyro.

### Slow-changing heading errors

Errors in “measuring” rate of turn with the virtual z-axis gyro can be divided roughly into two groups: (1) fast-changing errors, mostly from measurement noise, which have little effect in the heading computation, and (2) slow-changing errors, predominantly resulting from the combined drift of the physical gyros. Another source of slow-changing errors is the susceptibility of certain MEMS-based gyros to linear accelerations. This issue is particularly severe in the foot-mounted IMU of our PDR system, where accelerations due to gravity and foot-swing introduce substantial errors in the physical gyros. However, during walking at a constant pace and on level or fixed-slope terrain, these errors can be averaged for each step and remain about constant from one step to the next. The errors vary when users change pace or when the terrain slope changes. After these events, the errors stay fairly constant again. We can therefore regard them as slow-changing errors, just like drift. In the context of this paper, we lump all slow-changing errors together and refer to them collectively as “drift.” If we were able to estimate drift exactly and subtract it from the measurements of the virtual z-axis gyro, then the remaining fast-changing errors would average around zero, and, after integration, would not cause any significant heading errors. As we will see, our proposed method does exactly that: it estimates slow-changing errors and corrects for them.

We found these two abstractions to work very well in practice. They allow us to apply our drift correction method (discussed in the following section) without having to take into account the physical properties of the moving agent. As a result, our method applies equally well to walking persons, mobile robots equipped with 6-DOF inertial navigation systems, and vehicles equipped with a single-axis gyro or even just odometry (operating on flat or moderately rolling terrain).

## **2.1 Heading estimation**

Suppose a person equipped with a virtual Z-axis gyro was walking straight forward. Moving straight forward, the output of that gyro should be exactly zero throughout the walk. However, due to a variety of slow-changing errors the actual output is off by some small value  $\varepsilon$ . Suppose further that we divide the total travel distance into smaller intervals. A natural choice for the length of an interval in a pedestrian tracking application is from footfall to footfall. We define “footfall” as a single instance in which the IMU-equipped foot is firmly in contact with the ground and the forward velocity of the shoe at the contact point is zero. Since only one foot is instrumented, there are two steps between footfalls.

Due to the drift error  $\varepsilon_d$ , in each interval the rate of turn computed based on the z-axis gyro is

$$\omega_{meas,i} = \omega_{true,i} + \varepsilon_{d,i} \quad (1)$$

where

$\omega_{meas,i}$  – Rate of turn around the world z-axis during sampling interval  $i$ .

$\omega_{true,i}$  – True but unknown rate of turn around the world z-axis during sampling interval  $i$ .

$\varepsilon_{d,i}$  – This is the sum of all unknown, slow-changing errors during sampling interval  $i$ .

The change of heading in sampling interval  $i$ , denoted  $\Delta\psi_i$ , is computed by numerically integrating  $\omega_{meas,i}$

$$\Delta\psi_i = \omega_{meas,i} T_i \quad (2)$$

where

$T_i$  – Duration of time interval  $i$  in [sec].  $T_i$  is the time between footfall  $i-1$  and footfall  $i$ .

and the new estimated heading is

$$\psi_i = \psi_{i-1} + \Delta\psi_i \quad (3)$$

It is apparent from the simple Equations (1) through (3) that the estimated heading,  $\psi_i$ , represents the true heading and the accumulated sum of all errors. The proposed FDHC algorithm makes the heuristic assumption that the follower has generally the same heading as the leader except when turning corners. In most urban and rural environments, this assumption is true *most* of the time. The strength of the FDHC algorithm is that it corrects drift errors whenever the heuristic assumption is true, and it makes only small or no corrections at all when the assumption is false.

## 2.2 The FDHC Algorithm

In general terms, FDHC works as follows. After each footfall, FDHC compares the follower's current heading,  $\psi_{F,i}$ , with the heading that the leader had at the waypoint (i.e., footfall) that is closest to the follower's current position. That waypoint, which was recorded  $n$  footfalls ago, is labeled  $\psi_{i-n}$ . FDHC assumes that any difference between the two headings represents the accumulated heading error of the leader, caused by gyro drift  $\varepsilon_d$ . FDHC then issues a small, fixed-magnitude correction based on the presumed *sign* of the error. Subsequently, the correction is added to an accumulator, the content of which is subtracted from all future  $\psi_i$ . If after the next footfall the sign of the accumulated error is still the same (although the magnitude of the error was reduced), then FDHC issues another correction that is added to previous corrections. This process is repeated at every footfall, until the sign of the perceived error is reversed.

FDHC therefore works by issuing small corrections repeatedly, not by issuing a single large correction.

In the following sections we explain in detail how the FDHC algorithm:

- models drift  $\varepsilon_d$  as a disturbance in a feedback control system;
- estimates the magnitude of this disturbance by examining the content of the accumulator in the I-controller of that feedback control system;
- remains largely insensitive to changes in  $\psi$  that have large-amplitudes but short duration.

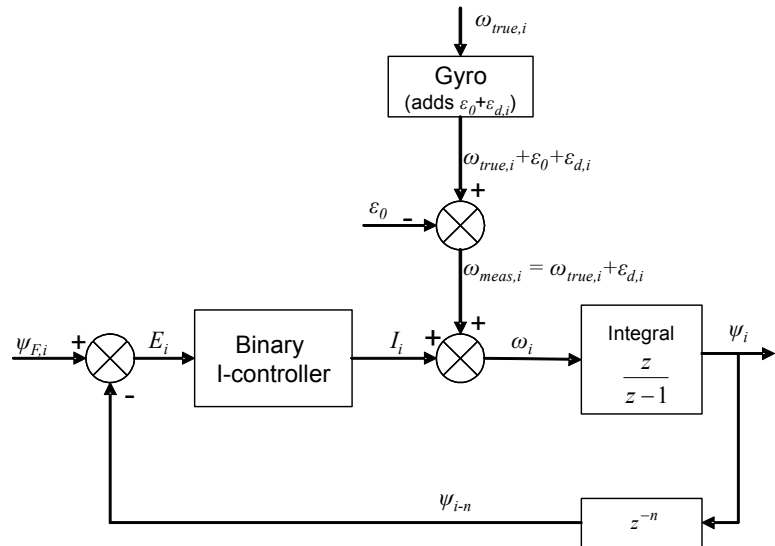
The FDHC algorithm functions essentially like a feedback control system. This is different from most other measuring systems, where signals pass from the sensor to the instrument's output in open-loop fashion. Figure 2 shows a block diagram of the feedback control system for the FDHC algorithm.

We start the explanation of the feedback control system with the signal from the gyro, which is modeled as a disturbance in the block diagram of Figure 2. For the purpose of explaining the feedback control system, let us assume that the leader is traveling straight ahead along the predicted direction. We will discuss later how the algorithm handles cases when this assumption is not true. When walking straight,  $\omega_{true} = 0$ , so the only output from the gyro is drift,  $\varepsilon_d$ .

This signal is added to the output of the binary I-controller, which will be explained later in this section. Initially, the output of the I-controller is zero, so  $\varepsilon_d$  is passed through to a numeric Integrator, which computes the relative heading,  $\psi_i$ . The block labeled  $z^{-n}$  is a pure delay of  $n$  sampling intervals. For now, let us assume  $n = 1$ .

After the first iteration, when  $i > n$ , the control loop can be closed by submitting the previous value of  $\psi$ ,  $\psi_{i-n}$ , to the comparator.  $\psi_F$  corresponds to the follower's current heading, which we compare to the leader's heading.  $E$  has a positive, negative, or zero value if the predicted heading lies clockwise from, counterclockwise from, or in the same direction as the heading we compare it to, respectively.

This brings us to the binary I-controller. Unlike conventional integral (I) or proportional-integral (PI) controllers, the binary I-controller is designed not to respond at all to the magnitude of  $E$ . Rather, it only responds to the sign of  $E$ . If  $E$  is positive (i.e., heading points to the



**Figure 2:** The HDE algorithm viewed as a feedback control system. The block labeled “Binary I-controller” is explained in the narrative.

left of the predicted direction), then a counter (called “Integrator” or “ $I$ ”) is incremented by a fixed small increment,  $i_c$ . If  $E$  is negative (i.e., heading points to the right of the predicted direction), then  $I$  is decremented by  $i_c$ . In this fashion, and although the controller does not respond to the magnitude of  $E$  immediately, *repeated* instances of  $E$  having the same sign will result in repeated increments or decrements of  $I$  by  $i_c$ .

The reason for defining a *binary* I-controller is that the ideal condition of follower heading  $\Psi_{F,i}$  being perfectly aligned leader heading  $\Psi_{i-n}$  is rarely met. Indeed,  $\Psi_{F,i}$  can differ significantly from  $\Psi_{i-n}$ , such as when the leader has already turned around a corner while the follower has not, or when the follower circumnavigates an obstacle that the leader stepped over. In that case a conventional I-controller does not work well since it would respond strongly to the large value of  $E$ , even though this is not necessarily an indication of a large amount of drift. The proposed binary I-controller, on the other hand, is insensitive to the magnitude of  $E$ . Instead, the controller reacts slowly to  $E$  having the same sign *persistently*.

We can now formulate the binary I-controller

$$I_i = \begin{cases} I_{i-1} - i_c & \text{for } E < 0 \\ I_{i-1} & \text{for } E = 0 \\ I_{i-1} + i_c & \text{for } E > 0 \end{cases} \quad (4a)$$

where

$i_c$  – fixed increment, also considered the gain of the binary I-controller in units of degrees.

An alternative way of writing Eq. (4a) is

$$I_i = I_{i-1} + \text{SIGN}(\Psi_{F,i} - \Psi_{i-n})i_c \quad (4b)$$

where SIGN() is a programming function that determines the sign of a number. SIGN( $x$ ) returns ‘1’ if  $x$  is positive, ‘0’ if  $x = 0$ , and ‘-1’ if  $x$  is negative.

The next element in the control loop adds the controller output to the raw measurement

$$\psi_i = \psi_{i-1} + \Delta\psi_i + I_i \quad (5)$$

As long as  $i_c$  is of greater magnitude than typical increments in drift errors, the Integrator continuously tracks the heading error (but with an opposite sign), just like the Integrator in a conventional I-controller tracks slow-changing disturbances. At steady state, i.e., when  $\Psi_{F,i}$  is mostly aligned with  $\Psi_{i-n}$  (as is typically the case when the leader and the follower travel along a straight corridor or road) the content of the Integrator will oscillate about  $-\varepsilon_d$  with an amplitude of  $i_c$ , which is typically much smaller than  $1^\circ$ . The strength of the FDHC algorithm lies in the fact that it can readily tolerate short deviations from walking in the same direction as the follower. For example, when a robot follower avoids a local obstacle that the human leader simply stepped over or when turning, large discrepancies between the leader and the follower heading are incurred, but only for a short period of time. As a result, FDHC may increment or decrement the Integrator in the wrong direction – but only for the short duration of the discrepancy. However, once the headings of the leader and the follower are roughly aligned again, FDHC resumes the correction of drift errors.

### 2.3 Waypoint Matching

In the ideal case of the follower following the leader’s trail exactly, both the leader and the follower should have the same heading as they pass through the same waypoint. Any difference in heading is interpreted as being the result of drift in the leader’s gyro. Waypoints are defined by the leader’s footfalls, which are typically between 1.0 and 1.5 meters apart from each other. There are thus a large number of waypoints generated during a walk. One problem is that the follower reaches each waypoint with a lag of a few or even tens of seconds.

Ideally, the FDHC algorithm should compare the follower’s heading at or near waypoint  $i-n$  (where  $i$  is the leader’s most recent waypoint and  $n$  is an index into the list of waypoints) to the heading that the leader had when passing through waypoint  $i-n$ ,  $n$  footfalls ago. To do so, the follower transmits its momentary position  $P_F$  and heading  $\psi_F$  at footfall intervals. The leader receives this data and finds the point  $P_L$  in the leader’s breadcrumb trail that is closest to  $P_F$ .

Unfortunately, we found that this approach introduces a number of problems. First, there is an issue of incorrect correlation of points around self-intersection of the path. We were able to resolve this problem by considering the gap in se-

quence number between the subsequent selected points. A bigger problem is that this approach depends on accurate position information to produce accurate heading data. While the leader's position errors are substantially reduced by correcting the leader's heading, they are not eliminated and will continue to grow without bound. In walks lasting tens of minutes or more (depending strongly on the curviness of the path and other factors), this will inevitably lead to incorrect correlation of points, and therefore heading errors cannot be said to be bounded.

## 2.4 Future improvements

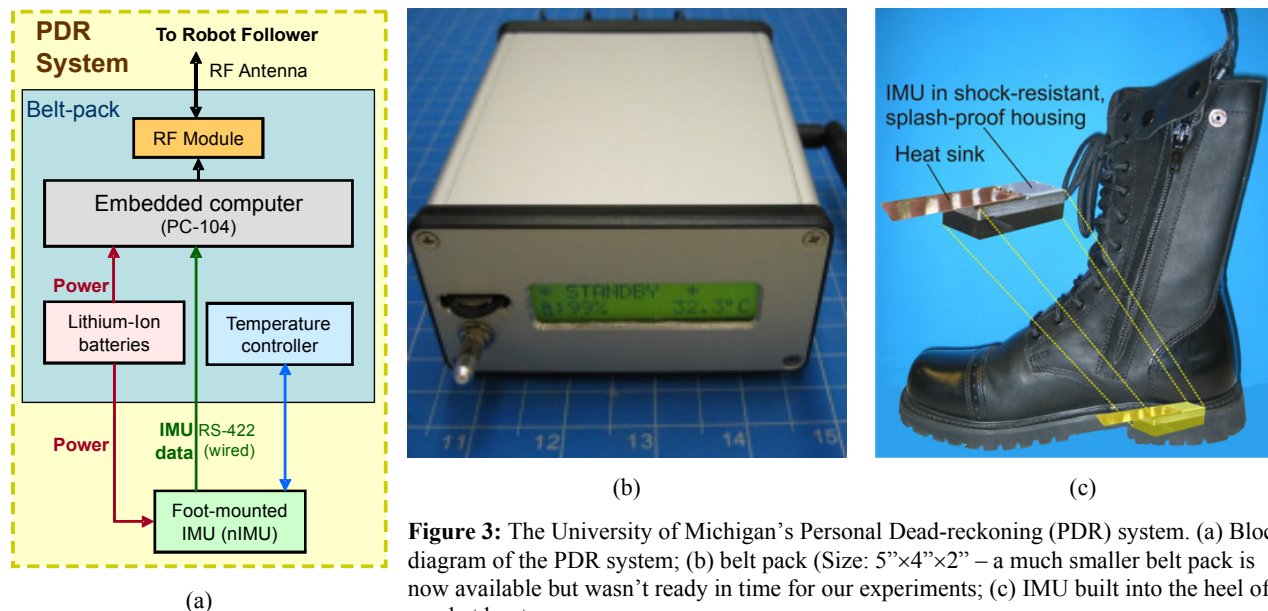
Our ultimate goal is not just correcting the leader's heading errors (which is the limited objective of the work described in this paper), but also to correct the leader's position errors. We think that in order to achieve this ultimate goal in missions of unlimited duration, there must be some means for at least occasionally (e.g., once every few minutes) measuring the absolute lag, in terms of time or distance, between the follower and the leader. We further think that this can be accomplished very well by considering the series of changes in heading that the leader experiences during a walk as a stream of features that can be unambiguously identified by the follower by means of a Particle Filter. Dean et al. [7] demonstrated excellent results using pre-mapped road pitch and road roll as the only features fed to a particle filter for a car traveling on that road. Changes in heading are much more distinct features that should therefore be easy to match. Once the follower identifies a feature, it can compare the timestamp of that feature (i.e., the time at which the leader passed by this feature) to the time at which the follower passed that feature. Doing so establishes the absolute follower lag in terms of time,  $T_{lag}$ . The follower then adds the value of  $T_{lag}$  to the position and heading data packages that it transmits to the leader. This enables the leader to compare the follower's momentary heading to the leader's heading  $T_{lag}$  ago. The follower then keeps track of  $T_{lag}$  by monitoring its own and the leader's speed, at least until the next feature is matched. Of course, with the availability of feature-matching capabilities, it will also be possible to correct the leader's position errors.

## 3 THE EXPERIMENTAL SYSTEM

We tested the leader-follower system under real-world conditions. The tested system comprised two components: The University of Michigan's Personal Dead-reckoning (PDR) system, used by the human leader, and SPAWAR's robot follower. Before we present the experimental results in Section 4, we offer a brief description of both systems.

### 3.1 Personal Dead-reckoning (PDR) system

Figure 3a shows a block diagram of the PDR system, which we developed in earlier work at the University of Michigan [8][9]. The system comprises an Inertial Measurement Unit (IMU) and a belt-pack that holds a computer, Lithium-Ion battery pack, and custom electronics (shown in Figure 3b). The IMU can be strapped to the medial side of the user's



**Figure 3:** The University of Michigan's Personal Dead-reckoning (PDR) system. (a) Block diagram of the PDR system; (b) belt pack (Size: 5"×4"×2" – a much smaller belt pack is now available but wasn't ready in time for our experiments); (c) IMU built into the heel of a combat boot.

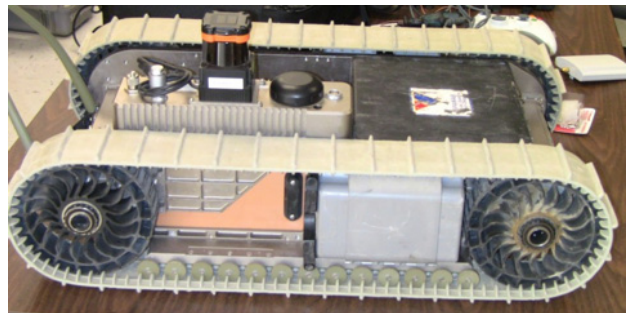
right shoe (to allow transferring the IMU from one user to another), or the IMU can be embedded in the heel of a boot, as shown in Figure 3c and d, respectively.

A foot-mounted IMU is somewhat of an inconvenience because of the need for power and data wires running up to the belt-pack. However, mounting the IMU on the foot allows the use of a method known as “Zero Velocity Update” (ZUPT) to compensate for drift in the accelerometers. ZUPT works for foot-mounted IMUs because during every foot-fall, when the foot is firmly on the ground, the velocity of the sole at that point is zero (or near zero). If velocity computed from the accelerometers is not equal to zero, then the accelerometers must be wrong and we can compute by how much the accelerometers are erring. With our implementation of ZUPT and some additional signal processing we achieve consistently errors of 1% of distance traveled. Systems similar to ours but developed independently are described in [10] and [11].

One problem with the PDR system is that we can apply ZUPT only to the accelerometers, but not to the gyros in the foot-mounted IMU. This is because ZUPT requires a reference signal one level of integration above the signal to be corrected. In the case of rates, this reference signal would be angle. It is not possible to measure gyro bias instability during footfalls either, because at footfall, the gyro is capable of registering small foot-ground interactions which in turn obscure the bias instability. With regard to tilt, we can bound roll and pitch errors by tilt data provided by the accelerometers during footfall. However for heading, there is no good method for estimating or otherwise reducing the effects of drift. It is this limitation of the IMU’s heading estimation that motivated the present work in the first place.

### 3.2 Test Platform and Software

The robot follower is an iRobot Packbot [6] with a computational payload that includes a 1.2 GHz dual core CPU and the following sensors: Hokuyo URG-30LX LADAR, Microstrain 3DM-GX1 IMU, Ublox GPS, and platform encoder data (see Figure 4). We also added a KVH DSP-3000 fiber optic gyro [12] to the system. The navigation solutions of the robot are based on an Adaptive Extended Kalman Filter (KF) [13]. The KF fuses odometry, IMU, and GPS data, and the Karto SLAM algorithm [14], which uses the output of the Kalman Filter and the LADAR data. In order to maintain a consistent position estimate depending on whether SLAM is enabled or not (it usually is turned off while outdoors), and to ensure that modules can utilize position estimates from multiple sources, a coordinate frame management module is used to keep track of the transforms between coordinate frames, and to determine the current best estimate of position. A 2-D point cloud (i.e., LADAR data) based pursuit algorithm (see Figure 5) and pursuit position control behavior is used to determine the initial location and coordinate frame transformation of the leader as well as follow-the-leader when they are in view [5]. A fuzzy logic based obstacle avoidance algorithm is used to arbitrate the output of the pursuit position control behavior in order to avoid obstacles while trying to maintain the desired distance from the leader [15]. These software modules are all part of the SPAWAR Autonomous Capabilities Suite (ACS) [16].



**Figure 4:** iRobot Packbot with single-wide computational payload used for testing.

## 4 EXPERIMENTAL RESULTS

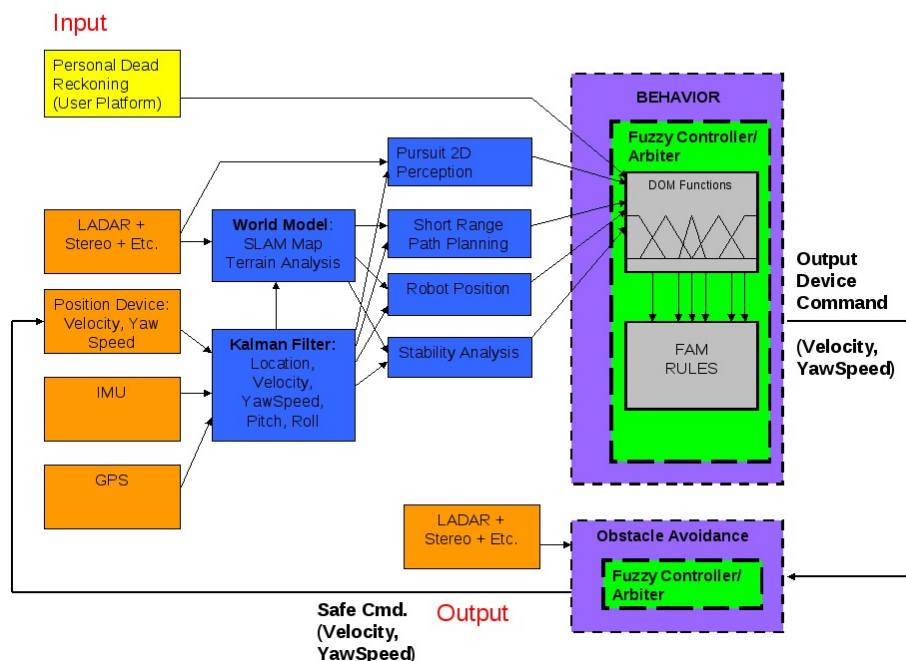
The leader-follower system was tested at SPAWAR San Diego. We performed several experiments, in which the leader-follower team moved along a very challenging path of mostly curving streets, some of which with steep slopes. Experiments took between 9-15 minutes each. The follower always used its onboard sensor to identify and track the leader, except when line-of-sight was briefly interrupted after the leader turned around a corner. The follower’s distance to the leader was always short, fluctuating between 0.5-2.0 m. As a result, direct line-of-sight outages were usually very short, on the order of 1-3 seconds. In every experiment there were several instances, in which the follower briefly diverted from the leader’s path, for example, to avoid obstacles. Since the follower almost always maintains line-of-sight, it would be trivial to correct the leader’s heading and position errors from the available measurements of distance and bearing to the leader. However since our work here is a precursor for extensive non-line-of-sight operation, we did not

use any of the follower's distance/bearing measurements to correct the leader's position or heading. Rather, we only used our proposed FDHC method, which does not require line-of-sight to correct the leader's heading.

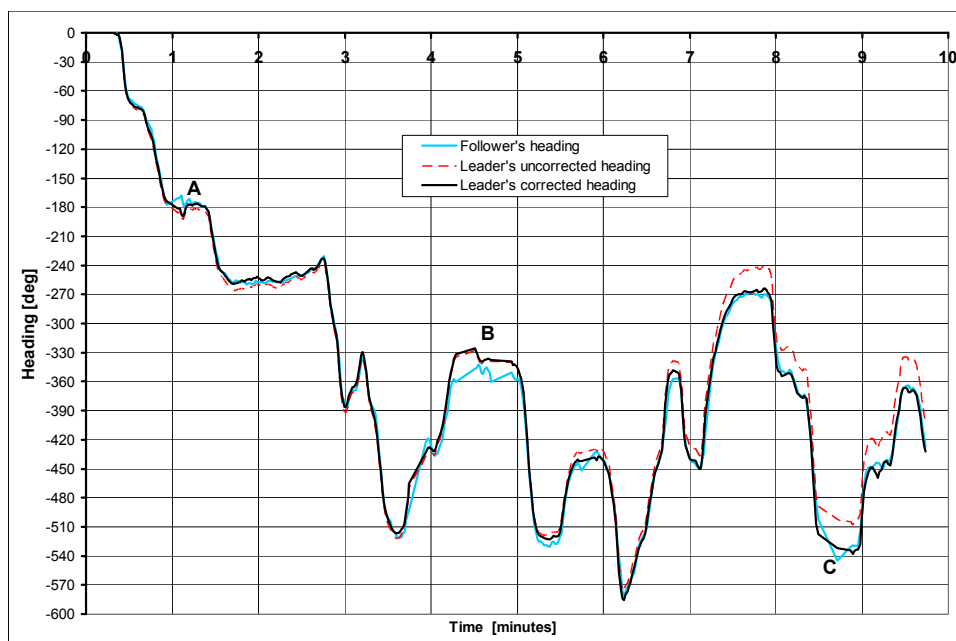
Figure 6 shows the result of a typical ~10-minute experiment. We plotted perpetual heading, that is, accumulated angles that are not normalized to a range within 0 to 360° or -180° to +180°. The follower heading is shown as the solid, light-blue curve. The dotted, red curve shows the leader's uncorrected heading as estimated by the PDR system. The solid, black curve shows the leader heading after correction with the FDHC algorithm. We recall that the purpose of the FDHC method is to correct the leader's heading errors caused by gyro drift, based on the follower-estimated heading. For that reason, the leader heading that coincides with the follower's heading is correct, as long as both team members traveled along the same path.

The exceptions are areas labeled 'A', 'B', and 'C', as well as some others. In these areas the follower temporarily diverted from the leader's path, usually to avoid obstacles. A particular strength of the FDHC algorithm is that it does not significantly affect the leader's heading while the follower is navigating around obstacles.

While subtle differences may be difficult to see in this plot, it is quite apparent that after about five minutes, uncorrected drift causes noticeable



**Figure 5:** Data flow diagram for behavior arbitration with the Pursuit Task in SPAWAR's robot follower.



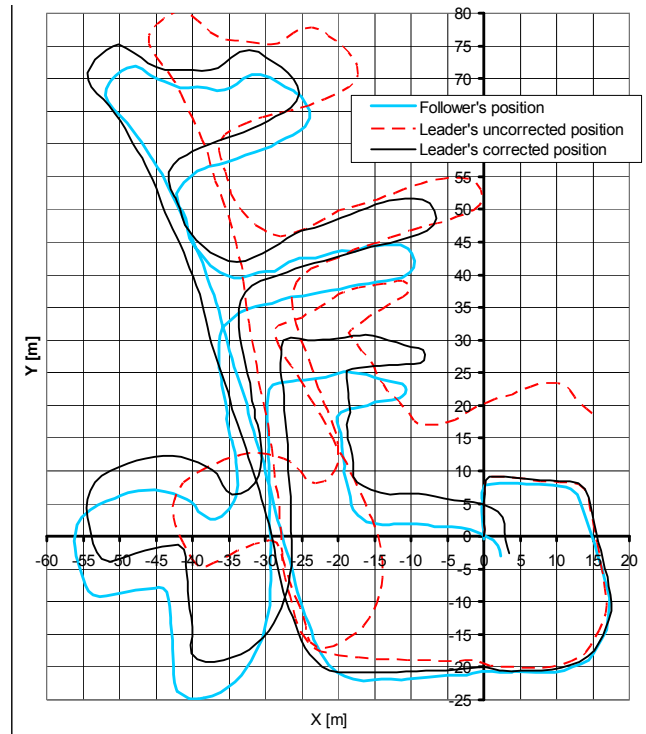
**Figure 6:** Result of a typical 10-minute leader-follower experiment. Light blue solid curve: the follower's perpetual heading (i.e., heading as computed from integrating yaw rate and not normalized to a range of -180° to +180°). Red dotted curve: uncorrected perpetual heading as estimated by the PDR system on the leader. Black solid curve: the leader's heading after correction with the FDHC algorithm. Note that leader data is plotted with a delay of two seconds (i.e., shifted to the right) so that sharp heading changes of the leader and the follower coincide in time.

heading errors for the leader. These errors grow to as much as  $30^\circ$  toward the end of this  $\sim 10$ -minute run. In contrast, FDHC corrections completely eliminate differences between the leader and the follower heading, in steady state. Given that ground truth (i.e., the follower's heading) is rather noisy, and since the follower's path often differs from the leader's path momentarily, it is difficult to show that the leader's corrected heading errors are indeed zero in steady state. However, by inspecting leader and follower trajectories plotted in Figure 7, one can see that relatively straight sections of leader and follower trajectories are indeed parallel to each other.

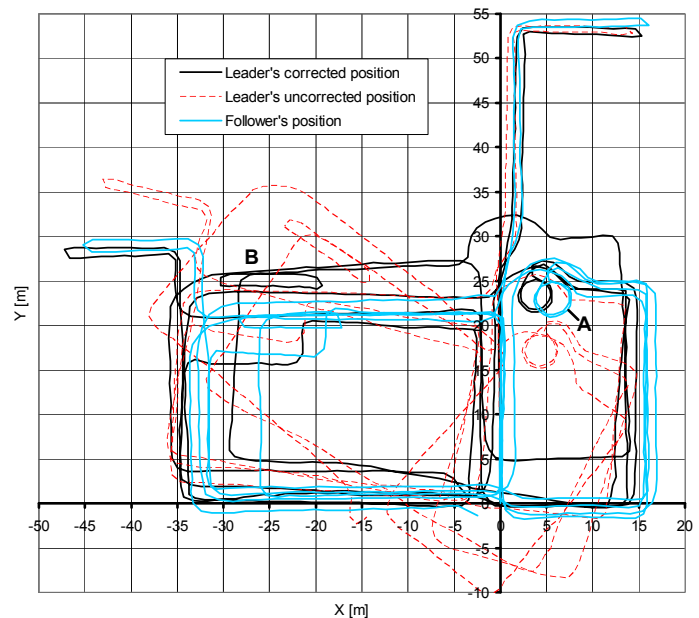
Unfortunately, we were unable to verify the ability of FDHC to correct the leader's heading errors from follower heading for follow distances larger than 2 meters with the robot follower. This is because the robot follower must maintain line-of-sight with the leader for successful tracking in cluttered environments. We did, however, perform some experiments in which a human follower followed a human leader. The human follower was also equipped with a PDR system, like the leader. However, the human follower's PDR system had a feature that allows very accurate heading estimation when used inside buildings. In the leader's PDR this feature was intentionally disabled. In this experiment, depicted in Figure 8, the follower distance and lag time varied from about 10 meters and 10 seconds, respectively, to as much as 50 meters and 50 seconds, respectively. The indoor terrain was favorable in that it comprised of mostly straight, long corridors. However, there were also substantial challenges, such as a spiral staircase (at A) and a switchback staircase (at B). In this and in all five similar indoor experiments that we conducted, the FDHC algorithm achieved zero heading errors at steady state. However, small heading errors could develop *temporarily* during the up to 50 seconds before corrections became available.

## 5 CONCLUSIONS

We developed and demonstrated a method for the correction of heading errors incurred by the leader in a leader-follower team. In our case the leader was a walking person and the follower was either a robot or another walking person. In either case, the follower was equipped with substantially more accurate position estimation instruments. While in our experiments the follower tracked the leader by means of line-of-sight sensors, this work was aimed at paving the way for teams that work completely without line-of-sight. To this end our proposed FDHC method is



**Figure 7:** Position plot of the  $\sim 10$ -minute experiment of Figure 6. Solid red line: follower; dotted blue line: leader, uncorrected; solid yellow line: leader, corrected. Since our algorithm only corrects leader heading but not leader position, small position errors develop from temporary heading errors.



**Figure 8:** Position plot of the 16-minute human leader-human follower experiment. The human follower lagged by up to 50 seconds and 50 meters. Solid light blue line: follower; dotted red line: leader, uncorrected; solid, black line: leader, corrected.

designed to work essentially without line-of-sight. The FDHC method uses accurate heading estimates from the follower to correct the leader's heading data. Our experiments show that FDHC works even if the follower lags tens of meters or tens of seconds behind the leader. The FDHC method effectively eliminates heading errors in steady state. If follower lag (i.e., distance or time) is known, missions can last indefinitely. If follower lag is not known, missions of a few tens of minutes are possible, before heading errors will grow without bound.

Limitations of the FDHC algorithm, to be addressed in future work, are: (1) only heading errors are corrected, but not position errors, and (2) follower lag (distance or time) must be known to allow missions of unlimited duration.

### Acknowledgements

This work was supported by the Center for Commercialization of Advanced Technology (CCAT) San Diego. Special thanks are due to Commander (Ret.) H.R. (Bart) Everett, Technical Director for Robotics at Space and Naval Warfare Systems Center Pacific (SPAWAR), San Diego, who first proposed the idea of using the accurate position estimation capabilities of a robot follower to correct dead-reckoning errors of a human leader.

## 6 REFERENCES

- [1] Cowan, N., Shakernia, O., Vidal, R., and Sastry, S., "Vision-based Follow-the-Leader," *Proc. 2003 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, (2003).
- [2] Ng, T.C., Ibañez-Guzmán, J., Shen, J., Gong, Z., Wang, H., Cheng, C., "Vehicle Following with Obstacle Avoidance Capabilities in Natural Environments," *Proc. IEEE Int. Conf. on Robotics & Automation*, (2004).
- [3] Mariottini, G.L., Pappas, G., Prattichizzo, D., Daniilidis, K., "Vision-based Localization of Leader-Follower Formations," *Proc. 44th IEEE Conf. on Decision and Control, and the European Control Conference*, (2005).
- [4] Hogg, R.W., Rankin, A.L., Roumeliotis, S.I., McHenry, MC., Helmick, D.M., Bergh, C.F., Matthies, L., "Algorithms and Sensors for Small Robot Path Following," *Proc. IEEE Int. Conf. on Robotics & Automation*, (2002).
- [5] Kogut, G., Ahuja, G., Sights, B., Pacis, E.B., and Everett, H.R., "Sensor Fusion for Intelligent Behavior on Small Unmanned Ground Vehicles," *Proc. SPIE 6561*, (2007).
- [6] iRobot, "PackBot," <http://www.irobot.com>, (2010)
- [7] Dean, A.J., Martini, R.D., and Brennan, S.N., "Terrain-Based Road Vehicle Localization Using Particle Filters," *2008 American Control Conference*, Seattle, Washington (2008).
- [8] Ojeda, L., and Borenstein, J., "Non-GPS Navigation for Emergency Responders," *2006 International Joint Topical Meeting: Sharing Solutions for Emergencies and Hazardous Environments*. Salt Lake City, UT (2006).
- [9] Ojeda, L. and Borenstein, J., "Non-GPS Navigation for Security Personnel and Emergency Responders," *Journal of Navigation* 60(3), 391-407 (2007).
- [10] Foxlin, E., "Pedestrian Tracking with Shoe-Mounted Inertial Sensors," *IEEE Computer Graphics and Applications*, 25(6), 38-46 (2005).
- [11] Yun, X., Bachmann, E.R., Moore IV, H., and Calusdian, J., "Self-contained Position Tracking of Human Movement Using Small Inertial/Magnetic Sensor Modules," *Proc. IEEE Int. Conf. on Robotics & Automation*, (2007).
- [12] KVH, "DSP-3000 fiber optic gyro," <http://www.kvh.com>, (2010).
- [13] Pacis, E.B., Sights, B., Everett, H.R., and Ahuja, G., "An adaptive localization system for outdoor/indoor navigation," *Proc. SPIE 6230*, (2006).
- [14] Karto, "SLAM algorithm," <http://www.kartorobotics.com/>, (2010).
- [15] Sights, B., Ahuja, G., Kogut, G., Pacis, E.B., Everett, H.R., Fellars, D., and Hardjadinata, S., "Modular Robotic Intelligence System Based on Fuzzy Reasoning and State Machine Sequencing," *Proc. SPIE 6561*, (2007).
- [16] Ahuja, G., Fellars, D., Kogut, G. T., Pacis, E. B., Sights, B., and Everett, H. R., "Test Results of Autonomous Behaviors for Urban Environment Exploration," *Proc. SPIE 7332*, (2009).